1. **Explain blue-green deployment in microservices.**

Blue-green deployment in microservices involves running two identical environments - one serves production traffic (blue), while the other hosts a new version (green). Traffic is gradually switched from blue to green, allowing for seamless updates with minimal downtime. If issues arise, traffic can be rerouted back to the stable blue environment, ensuring reliability and enabling quick rollback if necessary

2. ***What is the role of a container orchestration tool?***

Container orchestration tools like Kubernetes automate the deployment, scaling, and management of containerized applications.

3. ***How do you design microservices for resiliency in the face of network failures?***

Microservices should be designed to handle network failures gracefully, utilising patterns like Circuit Breaker, Retry, and implementing fallback mechanisms.

4. ***How do you design microservices to scale effectively and ensure optimal performance under varying loads?***

To scale microservices effectively, utilize horizontal scaling by deploying multiple instances of each service. Design services to be stateless and leverage containerization for easy replication. Implement load balancing to distribute traffic evenly, and employ caching mechanisms to reduce database load. Monitor system metrics and auto-scale based on demand.

5. ***Why are Container used in Microservices?***

Containers are easiest and effective method to manage the microservice based application. It also helps you to develop and deploy individually. Docker also allows you to encapsulate your microservice in a container image along with its dependencies. Microservice can use these elements without additional efforts.

6. ***What are the three commonly used tools for Microservices?***
   Following are the three commonly used tools for Microservices:
   - *Wiremock*
   - *Docker*
   - *Hysrix*

7. ***What are the three types of tests used in Microservices?***

   - ***Bottom Level Test:*** *The bottom-level tests perform general tests such as performance tests and unit tests. These kinds of tests are entirely automated.*

- ***Middle-Level Tests:*** *The middle-level tests are used to perform exploratory tests such as the stress test and usability test.*
- ***Top Level Tests:*** *The top-level tests are used to conduct acceptance tests, mostly fewer in numbers. These types of tests make stakeholders know about different software features.*

8. What is the main difference between SOA and the Microservices Architecture?

SOA stands for Service Oriented Architecture. It is a collection of services used to communicate with each other through simple data passing or activity coordination. On the other hand, the Microservices Architecture is a collection of small functional modules that are independently deployable, scalable, target specific business goals, and communicate over standard protocols.

9. What do you understand by Domain-Driven Design?

Domain-Driven Design is an architectural style based on Object-Oriented Analysis Design concepts and principles. It is used to develop a complex system by connecting the related components of the software system into a continuously evolving system. Domain-Driven Design is based on three principles:

- *Focus on the core domain and domain logic.*
- *Base complex designs on models of the domain.*
- *Collaborate with the domain experts to improve the application model and resolve any emerging domain-related issues regularly.*

10. What is the use of Bounded Context in Domain-Driven Design?

The Bounded Context is a central pattern in Domain-Driven Design. It is the core of Domain-Driven Design's strategic design section, which deals with large models and teams. It is used to divide the large models into different Bounded Contexts and being explicit about their inter-relationships.

11. What do you understand by Canary Releasing?

Canary releasing is a technique used to introduce new software versions by rolling out the updated version or new code/features to a subset of users as an initial test before making the entire infrastructure available to everybody. This technique is called canary release because it is based on canary releases in coal mines to alert miners when the toxic gases reach dangerous levels.

12. What is the difference between Monolithic, SOA and Microservices Architecture?

- ***Monolithic Architecture:*** *In this type of architecture, different components of an application like UI, business logic, data access layer are combined into a single platform or program.*

- ***SOA (Service Oriented Architecture):*** *In this architecture, individual components are loosely coupled and perform a discrete function. There are two main roles – service provider and service consumer. In SOA type, modules can be integrated and reused, making it flexible and reliable.*
- ***Microservices Architecture:*** *It is a type of SOA in which a series of autonomous components are built and combined to make an app. These components are integrated using APIs. This approach focuses on business priorities and capabilities and offers high agility, i.e. each component of the app can be built independently of the other.*

13. Why would you need reports & dashboards in microservices?

Reports and dashboards are mainly used to monitor and upkeep microservices. Multiple tools help to serve this purpose. Reports and dashboards can be used to:

- *find out which microservices expose what resources.*
- *find out the services that are impacted whenever changes in a component occur.*
- *provide an easy point that can be accessed whenever documentation is required.*
- *review the versions of the components that are deployed.*
- *To obtain a sense of maturity and compliance from the components.*

14. How does PACT work?

PACT is an open-source testing tool. It helps in testing the interactions between consumers and service providers. It offers support for numerous languages, such as Ruby, Java, Scala, .NET, JavaScript, and Swift/Objective-C.

It works through the following key principles:

- *Consumer-Driven Contracts*
- *Consumer Testing*
- *PACT file generation*
- *Services' Mocking and Testing*
- *Integration into CI/CD*
- *Compatibility between the consumer and the provider*

15. What are coupling and cohesion?

Coupling and Cohesion are fundamental concepts in software architecture. Coupling refers to the degree of interdependence between software modules. For example, a lower coupling means that each component is independent and changes in one will not affect the others. On the other hand, Cohesion refers to how closely the responsibilities of a module or component are related to each

other. For example, higher cohesion means that each module is responsible for a single task or function.

A well-designed application should aim for low coupling and high cohesion. Low coupling ensures that each service operates independently, minimizing the knowledge they need about each other. This independence makes the system more robust and easier to maintain. High cohesion means that all the related logic is kept within a single service, preventing unnecessary communication between services which could impact performance.

16. ***What is OAuth?***

Open Authorization Protocol, otherwise known as OAuth, helps to access client applications using third-party protocols like Facebook, GitHub, etc via HTTP. You can also share resources between different sites without the requirement of key-value credentials.

OAuth allows the account information of the end user to be used by a third party like Facebook while keeping it secure (without using or exposing the user's password). It acts more like an intermediary on the user's behalf while providing a token to the server for accessing the required information.

17. ***What are some of the common mistakes made while transitioning to microservices from monolithic architecture?***

Transitioning from a monolithic architecture to microservices can be complex, and several common mistakes might occur during this process:

- *Insufficient consideration of data management, testing, communication, and security aspects, leading to operational complexities and inconsistencies.Often the developer fails to outline the current challenges.*
- *Failing to adopt suitable DevOps practices and robust Continuous Integration/Continuous Deployment (CI/CD) pipelines*
- *Responsibilities, timeline, and boundaries not clearly defined.*
- *Failing to implement and figure out the scope of automation from the very beginning.*
- *Overlooking security considerations, such as authentication, authorization, and data encryption*

18. ***What do you mean by Bounded Context?***

In the context of Domain-Driven Design (DDD), a Bounded Context refers to a specific boundary within which a particular domain model is defined and applicable. DDD works with large models by disintegrating them into multiple bounded contexts. While it does that, it also explains the relationship between them explicitly. It represents a cohesive area or segment of a larger business domain where a set of closely related terms, concepts, rules, and interactions exist and are consistent.

19. ***What is the difference between REST and Microservices?***

- *REST is a means to implement Microservices.*
- *It is an architectural style that can develop large-scale applications that can be scaled up quite easily.*
- *It is used in API design as well as in web apps.*
- *We need to follow particular patterns to make microservices loosely coupled.*
- *Through REST we can build Microservices.*